IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 14, 2020, accepted April 28, 2020, date of publication May 15, 2020, date of current version June 3, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2994931

TIDCS: A Dynamic Intrusion Detection and Classification System Based Feature Selection

ZINA CHKIRBENE[1], AIMAN ERBAD[1], RIDHA HAMILA[1], AMR MOHAMED[1], MOHSEN GUIZANI[1], AND MOUNIR HAMDI[2]

[1]College of Engineering, Qatar University, Doha, Qatar
[2]College of Science, Engineering and Technology, HBK University, Doha, Qatar

Corresponding author: Zina Chkirbene (zina.chk@qu.edu.qa)

This work was supported by the NPRP Award from the Qatar National Research Fund (a member of The Qatar Foundation) under Grant NPRP 8-634-1-131. The statements made herein are solely the responsibility of the author[s].

ABSTRACT Machine learning techniques are becoming mainstream in intrusion detection systems as they allow real-time response and have the ability to learn and adapt. By using a comprehensive dataset with multiple attack types, a well-trained model can be created to improve the anomaly detection performance. However, high dimensional data present a significant challenge for machine learning techniques. Processing similar features that provide redundant information increases the computational time, which is a critical problem especially for users with constrained resources (battery, energy). In this paper, we propose two models for intrusion detection and classification scheme Trust-based Intrusion Detection and Classification System (TIDCS) and Trust-based Intrusion Detection and Classification System- Accelerated (TIDCS-A) for secure network. TIDCS reduces the number of features in the input data based on a new algorithm for feature selection. Initially, the features are grouped randomly to increase the probability of making them participating in the generation of different groups, and sorted based on their accuracy scores. Only the high ranked features are then selected to obtain a classification for any received packet from the nodes in the network, which is saved as part of the node's past performance. TIDCS proposes a periodic system cleansing where trust relationships between participant nodes are evaluated and renewed periodically. TIDCS-A proposes a dynamic algorithm to compute the exact time for nodes cleansing states and restricts the exposure window of the nodes. The final classification decision for both models is estimated by incorporating the node's past behavior with the machine learning algorithm. Any detected attack reduces the trustworthiness of the nodes involved, leading to a dynamic system cleansing. An evaluation of TIDCS and TIDCS-A using the NSL-KDD and UNSW datasets shows that both models can detect malicious behaviors providing higher accuracy, detection rates, and lower false alarm than state-of-art techniques. For instance, for UNSW dataset, the accuracy detection is 91% for TICDS, 83.47%by using online AODE, 88% for CADF, 90% for EDM, 90% for TANN and 69.6% for NB. Consequently, TICDS has better performance than the state of art techniques in terms of accuracy detection, while providing good detection and false alarm rates.

INDEX TERMS Cloud security, node past behavior, feature selection, trustworthiness, system cleansing, machine learning techniques.

I. INTRODUCTION

Cloud computing offers a reliable and cost-efficient model to provide internet-based services that are highly scalable 'as a service'. However, this model has several open issues that impact its credibility and applicability especially for dynamic networks, namely vehicular clouds and fog network [1].

In particular, the cloud applications require distribution, location-awareness, and mobility support, which increase the number of cyberattacks and complicate the trust situation [2]. For example, the rogue fog node is a node that pretends to have legitimate access and coaxes the network nodes to connect to it. In an insider attack (.i.e. where the attacker can steal property or information for a personal gain), a fog administrator, who can manage fog instances, can instantiate a rogue fog instance rather than a legitimate one. The existence of

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

fake fog node presents a big threat to personal data privacy and security [3]. Furthermore, the virtual machine instances are dynamically created/added/removed which makes it hard to maintain a blacklist of rogue nodes and to establish long-term trust between nodes. on the other hand, making a node online for extended periods can offer more time to attackers to explore and understand the server configuration. Therefore, a node that has been online and exposed to attacks can be assumed compromised. Despite the significant advantages of dynamic networks, they require a more demanding and dynamic resource constraint environment, which increases the security concerns and the number of attacks [4].

Several researchers proposed network intrusion detection systems (NIDS) to protect cloud environments from cyber-attacks.

IDS systems in IoT environment are extremely timely as we expect the unprecedented volume of attacks on various critical infrastructures [5]. The majority of proposed NIDS solutions are signature-based techniques that have some limitations [6]. For example, behavioral changes need to be easily detected, analyzed and attributable to specific elements of a network (e.g. operating system versions, protocols or individual users). However, the number of protocols and the diversity of data traversing through modern networks introduce high-levels of difficulty and complexity for NIDS in intrusion detection [7]. It increases the difficulty in establishing an accurate norm for attack detection where the state of an online node must be periodically evaluated to detect any abnormal behavior. Moreover, there are concerns related to some of these systems regarding the increasing levels of required human interaction which impact their efficiency.

Recently, machine learning techniques for intrusion detection have proven their efficiency. In [8], the authors proposed a new system for a secure network using machine learning techniques combined with the node's past behaviors. The use of past information about each participant node improves the trustworthiness in the network. However, the proposed model does not consider the fast change in node behavior, where the relationship between the participant nodes changes rapidly. The state of a node must be periodically evaluated with a periodic system cleaning. Another problem is the used datasets, which may include a great number of challenges such as the noisy data and the huge number of irrelevant features. The training and the validation of the learning model have to go through all these features increasing the computational complexity, the time consumption, and cause an overfitting issue [9]. These problems are critical for all applications as the systems need to rapidly detect any attack with a small run-time delay and low resource utilization. In particular, the limitations of the Internet of Things (IoT) systems make the previously mentioned problem more pronounced as the nodes need to consume a limited amount of resources while detecting attacks. To address the above limitations, the selection of significant features [10] while improving the detection accuracy is proposed. Many researchers studied the feature selection issue with high dimensionality problem, but generally, the proposed solutions suffer from an increase in the miss-detection rate [11].

In this paper, we propose two novel schemes called **Trust-based Intrusion Detection and Classification System (TIDCS)** (see figure 1) and **Trust-based Intrusion Detection and Classification System- Accelerated (TIDCS-A)** for secure network. The proposed systems introduce the idea of periodic system cleansing where trust relationships between participant nodes are evaluated and renewed periodically. The process involves removing irrelevant and redundant features utilizing a new algorithm for feature selection. The proposed algorithm generates the features subset randomly, which can reduce the time consumption compared to exhaustive and heuristic search by managing the number of iterations. Hence, multiple feature groups are selected randomly and the performance of the classifier is used as an evaluation criterion. The groups' selection is based on better exploitation of the best features that will be grouped together to be used by the machine learning algorithm. The use of a supervised machine learning algorithm is combined with the past information to improve the intrusion detection performance and the trustworthiness between the network nodes while avoiding the problems of low training data and dynamic behavioral node changes. TIDCS and TIDCS-A use the best-selected features to create a well-trained model used for an initial attack classification decision. Basically, each received packet goes through the trained model to get an attack classification decision. This decision is then saved in a secure database as node past information and explored during the final system classification. TIDCS applies periodic trust updates based on the node's past information. This period presents the number of considered past decisions called also Time Variable Status Unit (*TVSU*) and it is fixed depending on the network requirements (complexity, security level). Basically, in a dynamic network with massive data and high mobility, *TVSU* should have small values for a fast update. However, there are some standard closed cloud settings, namely private cloud, where

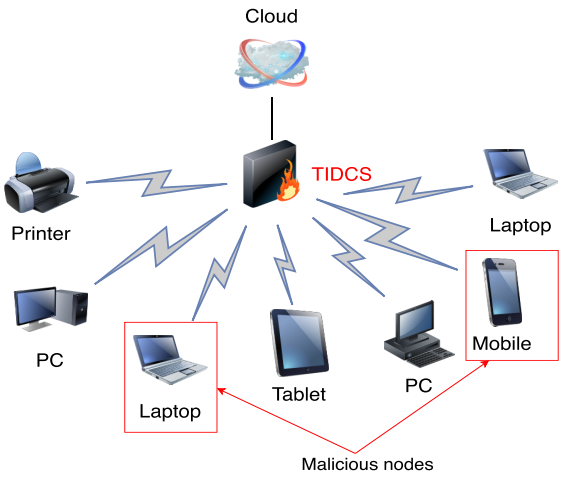

FIGURE 1. ***TIDCS* system model.**

nodes do not change their behaviors frequently, which causes the network to be mostly stable. In this case, *TVSU* can be relatively large. TIDCS-A proposes a dynamic algorithm to compute the exact time for nodes cleansing states and restricts the exposure window of the nodes called also the windows past decision size (*WPS*). The *WPS* is dynamically updated according to the node behavior to reduce the time it takes to detect the threat.

In the following, we highlight the major contributions proposed in this paper:

1) Designing a new algorithm for features selection using the random search combined with wrapper and filter techniques.
2) Designing a novel secure soft combination scheme based on machine learning algorithm and node behaviors to evaluate the trustworthiness of every node.
3) Developing a cleansing intrusion model using an automatic and periodic cleansing of the nodes based on their past behaviors.
4) Designing a new dynamic algorithm to determine the exposure window of the nodes according to its behavior.
5) Performing various simulations to evaluate the proposed solution, strengthen the theoretical analysis and compare the proposed model with the state-of-the-art techniques.

The rest of this paper is organized as follows: Section II provides a summary of the relevant work carried out in the area. Sections III, IV and VI describe the proposed schemes. Experimental results are presented in Section V. Section VII concludes the paper.

II. RELATED WORK

In this section, the proposed solutions for feature selection and network intrusion detection are presented.

A. FEATURE SELECTION METHODS

Feature selection and feature extraction are two general approaches for dimensionality reduction. Feature extraction methods transform existing features into a lower-dimensional space. During this process, new features are created based on linear or nonlinear combinations of features from the original set. Linear discriminant analysis (LDA) [12] and Principal Component Analysis (PCA) [13] are two popular techniques used for feature extraction and dimension reduction.

Feature selection is considered as a special case of feature extraction where the selection of a subset from the existing features is done without any transformation [14]. The feature selection techniques solve and minimize many problems that can be found in a typical machine learning problem such as noisy data and the huge number of irrelevant features. Feature selection is proposed in many works. In [15], the authors proposed feature selection methods based on mutual information. The optimal feature subset is defined using the relevance, redundancy and complimentary of the features. The idea is to find the feature subset that minimizes the cardinality while preserving the information contained in the whole set of features.

The LogitBoost-Based algorithm has been presented in [16] as an intrusion detection system using an ensemble classification approach called silent features. The proposed model reduces the number of features using both the filter and wrapper approaches to remove the redundant and irrelevant features. The LogitBoost-Based algorithm applies a heuristic search technique and chooses a genetic algorithm as the search function to define the relationship between the features. However, once the groups are generated, they will not be updated which may reduce the selection performance. The LogitBoost uses ensemble classification method based on a boosting algorithm. While boosting algorithm improves the classification performance, it is time consuming and computation expensive

Random Forests are popular to be used in feature selections [17] besides its main purpose for classification and regression. They provide an easy technique to rank the original set of features based on their ability to measure the importance score of each feature to obtain a subset whose performance is either equal or even better compared with the performance given by the complete original feature set. They are based on combining the idea of bagging with a random selection of features to build several decision trees and choose randomly at each node a subset of the features to split on.

The performance of the feature selection methods can be measured using many different metrics such as computer resources (memory and time), accuracy, the ratio of features selected, etc. According to [18], the evaluation of the produced subset can be done by filter and wrapper methods.

- Filter methods: Those methods evaluate the subset based on the uniqueness of the data using some statistical and ranking techniques that are independent of the used learning algorithm. This gave them the advantage of providing a subset that is created only once and can be used with different classifiers. Moreover, they are considered fast, efficient, less prone to overfitting and have a good generalization property. On the other hand, they do not consider the relationship between the different features and the performance of the subset varies from one used learning model to another. Some examples of filter-based algorithms are chi-squared, information gain, fast correlation-based filter, INTERACT, and a fast clustering-based feature subset selection [19].
- Wrapper methods: The evaluation of the subset depends on the used learning algorithm as it uses the performance accuracy of the classifier as an evaluation criterion and chooses the subset that gives the highest accuracy with the classifier. The advantages include having better performance and being able to consider the correlation between different features. On the other hand, they have a higher chance of being over-fitted and there is always a need to re-evaluate the feature selection process in case of using different learning algorithms. They are

classified into two types which are a sequential selection algorithms and heuristic search algorithms such as the Genetic Algorithms [19].

B. NETWORK INTRUSION DETECTION SYSTEMS

Intrusion Detection Systems based-signature play a crucial role in defending computer networks [20]. An IDS signature library has to be continually updated to detect the latest threats. Moreover, an IDS accuracy depends on the network address [21]. An attacker could falsify its IP address so that the IDS becomes unable to stop the intrusions to the network from taking place [22] which may reduce the attack detection efficiency. In [23], the authors presented the IDS as a combination of devices and software applications capable of detecting malicious activities and generating the corresponding report. In many cases, false positives reports are more frequent than actual threats [23]. So, the real attacks can slip through false reports or be ignored.

Collaborative anomaly detection framework (CADF) [24] comprises capturing and logging network data, preprocessing it to be handled at the decision engine sensor using the Gaussian Mixture Model (GMM) and interquartile range for identifying abnormal patterns. Moreover, the architecture for deploying this framework as Software as a Service (SaaS) is produced to be easily installed in cloud computing systems. The GMM can produce non-convex clusters that can be controlled with the variance of the distribution. However, GMM is not so trivial for optimizing the loss function, since it is not a convex function.

Euclidean Distance Map (EDM) for anomaly detection using sequential algorithms was presented in [25]. The system analyzes the network traffic and uses the distance maps to extract second-order statistics. These second-order statistics are exploited for new features generation which improves detection accuracy.

C. USE OF PAST INFORMATION

In [26], the authors proposed a weighted decision fusion scheme using past information. The model uses the local and global decisions of users to determine the reliability of each detector. However, these solutions do not take into consideration that principal-agent can be run by an untrusted service provider. To overcome this problem, the authors in [27] proposed that the users submit their encrypted data to the receiver which can only obtain the sum of the reports without learning each individual value. However, in the proposed model, the user location has to be continually updated for keys generation.

D. TRUST TECHNIQUES

In [28], the authors proposed a cluster and forward based on the trust cooperative spectrum sensing. The secondary users are divided into clusters and only the most trusted ones are selected for the sensing phase. The proposed solution reduces energy and delays transmission while improving the spectrum sensing performance. Also, the authors presented a solution for data injection attacks in [29] using trusted anchors detectors. It evaluates the instantaneous trustworthiness of mobile detectors based on reputation scores. However, this model requires a lot of resources to work (trusted anchors, GPS information) and it does not take into consideration the potential fusion centers' malicious behaviors.

E. USE OF MACHINE LEARNING TECHNIQUES

In general, intrusion detection can be approached by machine learning techniques which can be classified into three categories: unsupervised, supervised and hybrid machine learning techniques. An intrusion detection with a multi class SVM is presented in [30]. It increase the individual classification accuracy of the network attacks. In [31], an attack-resilient malicious node detection scheme is presented (BAN-Trust). This model can identify the malignant attacks on BAN according to the nature acquired through the nodes on their own and approvals shared by various nodes. In [31], the authors proposed a hybrid feature selection and two-level classifier ensembles are proposed. Features are selected based on the classification performance of a reduced error pruning tree (REPT) classifier. A new framework based on the organic integration of multiple deep learning techniques is proposed in [32]. A Damped Incremental Statistics algorithm is used to extract features from network traffic and train Autoencoder with a small amount of label data.

Triangle Area Based Nearest Neighbors (TANN) proposed in [33] as hybrid learning using unsupervised and supervised learning techniques. The k-means clustering is used first to obtain the center of the cluster of attack class. Then, the system computes the triangle area between the two centers to create a new feature signature of the data. Finally, the new feature is used by the k-NN classifier to improve the classification attacks. Triangle Area Based Nearest Neighbors is a **hybrid machine learning** technique that inherits the advantages of both the supervised and the unsupervised learning namely the good performance and unlabeled capability. However, the improvement in accuracy comes with high computation complexity and time consumption.

Authors in [34] proposed an online Naïve Bayes classifier for binary classification (2-classes normal / attacks) as well as multi-classification (23-classes) using the KDDCUP99 dataset. The proposed model tried to solve the issue of data changing in the network. However, it does not take into consideration the problem of time consumption as well as the data set imbalance. The classes having a high amount of data instances are most classified correctly while the classes with a low amount of data instances tend to be ignored.

Online Average One Dependence Estimator (AODE) model has been proposed in [35] for a multi-classification problem in the UNSW-NB15 dataset using a supervised machine learning algorithm. The proposed classifier updates data overtime to secure a dynamic network. The results show that the AODE outperformed Naïve Bayes (NB). In particular, the classification rates of AODE and NB are 83.47%

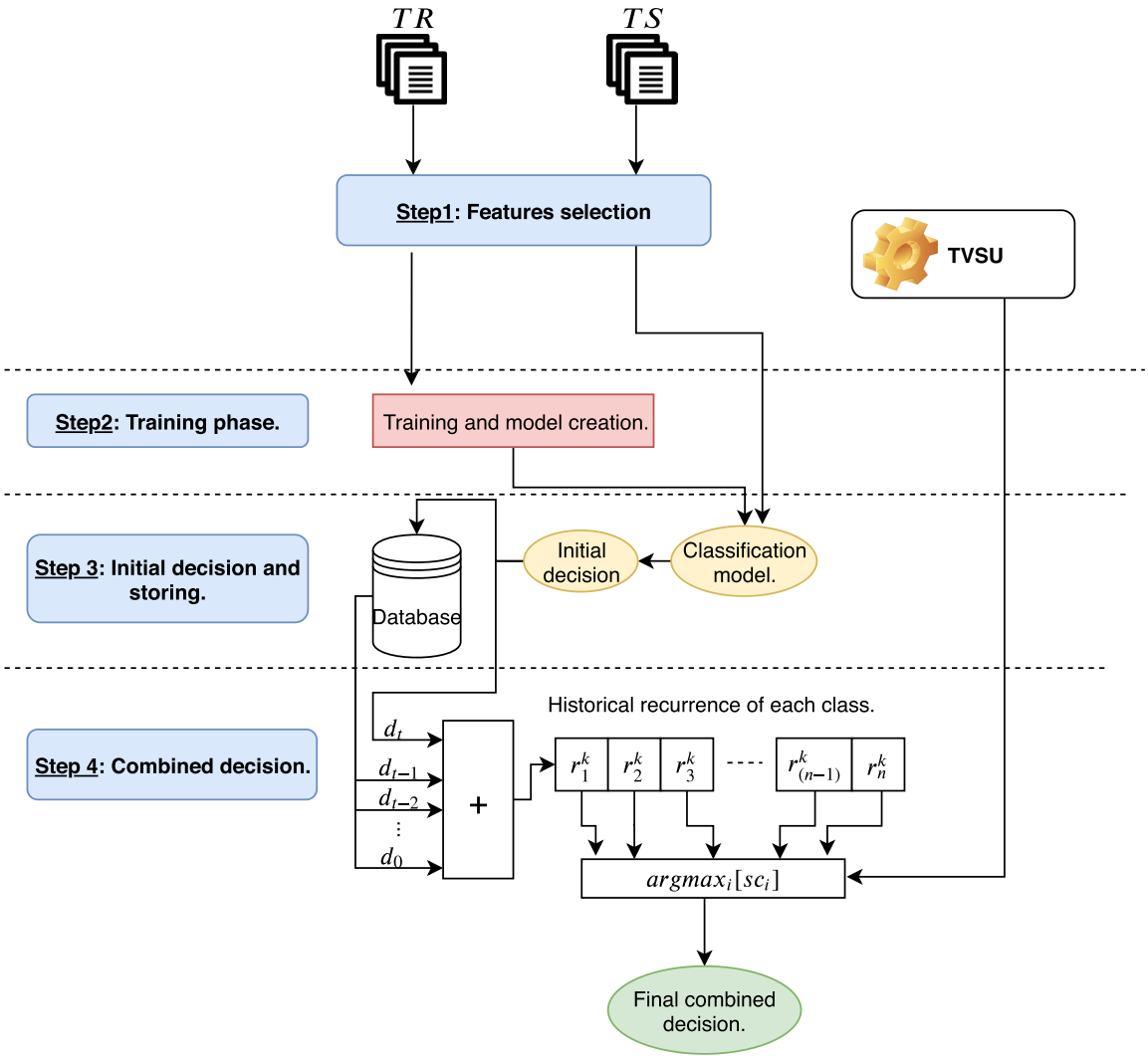

FIGURE 2. The general framework.

and 69.60% respectively for the multi-classification of the UNSW-NB15 dataset.

Both Naïve Bayes classifier (NB) and Online Average One Dependence Estimator (AODE) use the **supervised machine learning technique** which has relatively fast processing and high detection performance when compared to existing artificial neural networks and the unsupervised SVM [36]. However, maximizing the learning efficiency is still one of the major goals of secure systems that can be only achieved through a big amount of data, that might be challenging to collect.

III. THE OVERALL FRAMEWORK

Figure 2 shows the TIDCS framework. Firstly, the proposed system selects the important features from both the training and testing sets *TR* and *TS* respectively. Then, the set of the selected features are used as an input in the **Training phase** to build the classifier. For each received data point, the created classifier generates its decision and stores it in a private database during the **Initial decision and storing** phase. The final step related to **Combined decision** where the stored decisions in the database are preprocessed with the current classifier decision according to the used techniques (*TIDCS*/*TIDCS* − *A*). The main reason to consider the decision history is to detect malicious nodes with temporal characteristics during the classification process. In fact, multiple attacks are caused by the insertion of false information from compromised nodes within the network. So, the trust between nodes is needed in the network to ensure that the participating nodes are normal users and the use of past information to identify the network users' reliability can be a good approach to guarantee a trustworthy environment. In the proposed framework, the use of the supervised machine learning algorithm combined with the past information improve the trustworthiness between the network nodes and overcome the limited performance of the learning algorithm.

IV. FEATURE SELECTION

A. PROBLEM FORMULATION

To reduce time and complexity in the created machine learning model, TIDCS reduces the number of features to a subset of features called also best features. So, if we denote by $F = \{F_1, F_2, \ldots, F_{nf}\}$ the original set of features in the training data set denoted by *TR* with cardinality *nf*. *s* is the desired number of features per group *G* where $G \subseteq F$. In the presented system, the feature selection criterion function is the accuracy denoted by δ so the higher value of $\delta()$ indicates a better feature group. TIDCS aims to find a group $G \subseteq F$ so that:

$$|G| = s, \tag{1}$$

and

$$\delta(G) = \max_{z \subseteq F, |z|=s} \delta(z). \tag{2}$$

The proposed system completes the feature selection phase satisfying the conditions in Eq 2 based on two phases: random and improvement phases so that the best features are grouped together.

B. THE RANDOM PHASE

During the first *Ntraining* time slots, the groups *G* are selected randomly to increase the probability of making the features participate in multiple groups with different δ values allowing the system to distinguish the important ones faster. For each group, *G* with size *s*, TIDCS evaluates the set of features using a machine learning algorithm and computes $\delta(G)$. *Ntraining* and *s* are fixed according to the network requirement (complexity, time, accuracy). The proposed model computes the score of each feature participant in the group *G* and takes into consideration the number of times that the feature FE_i participates in the group generation.

C. IMPROVEMENT PHASE

After identifying the important features during the first *Ntraining* time slots, TIDCS selects the group to maximize the accuracy and applies a max-min strategy to select *s* the features with a high score in the same group. Figure 3 shows an example of the group generation of 15 features during the training and improvement phases where $s = 5$.

The feature selection process is done offline (during the learning phase). It can be controlled and can be tuned carefully because the scope of the training data is known. This does not impact the time complexity of the online classification process. Moreover, the overall training complexity can be control according to the *Ntraining* so that the bigger the *Ntraining* is, the more the training process will be.

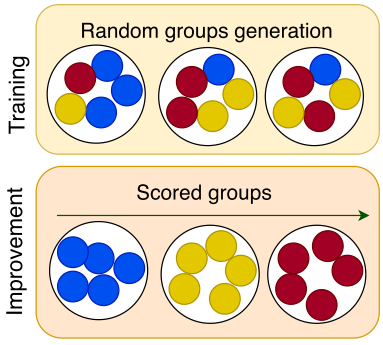

FIGURE 3. Features grouping during the random and improvement phases.

V. TRUST-BASED INTRUSION DETECTION AND CLASSIFICATION SYSTEM

Table 1 presents the notations used in the following sections.

TABLE 1. Table of notations.

Symbols	Meanings
R_t^k	The Historical recurrence vector
$\mathfrak{C}$	The possible classes in the dataset
M	Number of features .
WPS	The windows past decision size.
d_t^{*t}	the final decision at node k and time t.
N	Number of nodes in the network .
τ	Threshold value for the size of the window past decision fixed by the operator.
d_t^k	Classifier decision of received data from a node k in time t.
TVSU	The size of window past information for TIDCS .
γ_i	The time interval in index i.
s	The size of γ.
δ	The accuracy ratio.
μ	The accuracy correlation rate.
SMI	The similarity rate.
R_t^k	The historical recurrence vector.

A. TRAINING PHASE

We assume that the proposed network is composed of *N* nodes proportionally distributed according to the classes in the considered dataset.[1] We assume that each packet received in a node *k* at time *t* can be identified and differentiated by a set of features denoted by x_t^k (.i.e. IP destination, transmission protocol). Let y_t^k be the output class label for the k^{th} node at time *t*. We denote by *TR* the training input data extracted from the dataset. During the learning phase, the adopted machine learning algorithm (decision tree, SVM, random forest) generates an attack classification model by training the model with the available input data *TR*. This generated model uses the input features x_t^k to predict the outcome class y_t^k for any new node *k* at any time *t* so that:

$$y_t^k = classifier(x_t^k). \tag{3}$$

where $y_t^k \in \mathfrak{C}$ and $\mathfrak{C}$ denotes the set of all the attack class labels. Without loss of generality, $\mathfrak{C}$ is defined here by $\mathfrak{C} = \{0, 1, 2, 3, 4\}$ where 0 denotes a normally received data and 1..4 denote different attack types that will be detailed in the performance evaluation section.

[1]Note the distribution of the nodes categories does not affect the performance of the proposed model and is only used to exploit the available dataset to model a multimode continuous communication network.

B. CLASSIFIER DECISION AND STORING

For every newly received data at time t form a node k (every new entry extracted from the testing dataset TS), the classifier creates an initial decision d_t^k that does not depend on any past information. This decision is then stored in a secure database to increase data availability and reliability. In particular, the historical decision data should be secured to avoid any potential editing or injection of wrong information by malicious users to make the system unable to recognize their attacks. Also, this data should be private to make sure malicious users can not know and learn how the classification of the attack is working which is very crucial to the success of the proposed model.

C. COMBINED DECISION

The proposed model detects the intrusion based on the trust relationship between the nodes which is renewed periodically. Thus, TIDCS performs a regular update for the node behavior during each period $TVSU$ fixed according to the network requirements (network load/security level) and performance. Let $R_t^k = [r_1^k, r_2^k, .., r_n^k]$ be historical recurrence vector. It is considered as a summary of the decisions previously made by the classifier for the node k and stored in the database during $TVSU$. R_t^k is updated as follows:

$$\forall c_i \in \mathfrak{C}, R_t^k(c_i) = \left\{ \begin{array}{l} R_{t-1}^k(c_i) + 1 \text{ if } c_i = d_t^k, \\ R_{t-1}^k(c_i) \text{ if } c_i \neq d_t^k. \end{array} \right\} \quad (4)$$

In particular, $d_t^k \in \mathfrak{C}$ denotes the raw decision made at time t for node k and $R_t^k(c_i)$ counts the number of times the node k has been classified as c_i till the time t. From Eq. (4), it can be deduced that at time T, the number of times the classifier decided that the node k transmission was of class c_i is given by

$$r_i^k \triangleq R_t^k(c_i) = \sum_{t=TVSU}^{T} \left(d_t^k = c_i\right), \quad (5)$$

TIDCS extracts the most frequent decision in R_t^k during $TVSU$ i.e. the final decision at node k and time t is given by:

$$d^{*t}_k = \underset{c_i}{argmax}(R_k^t(c_i)). \quad (6)$$

Figure 4 shows an example of the most frequent decision for a node k with $TVSU = 3$, two classes c_0 and c_1 are considered. After 5 time slots, the number of times the node k was classified as c_0 and c_1 are $R_5^k(c_0) = 2$ and $\frac{k}{5}(c_1) = 1$, respectively. Therefore, the most frequent decision in this example is $d_5^{*k} = c_1$.

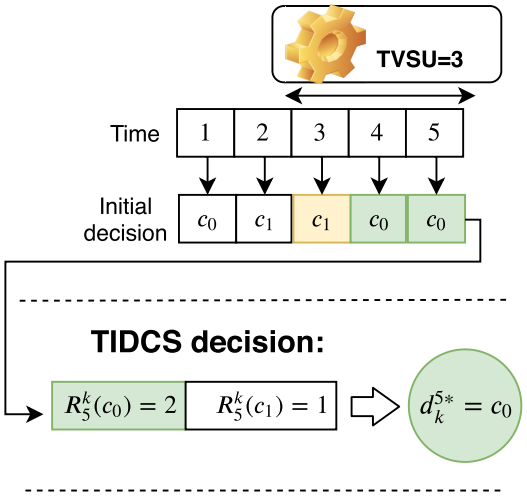

FIGURE 4. An example of the most frequent decision with $TVSU = 3$.

VI. TRUST-BASED INTRUSION DETECTION AND CLASSIFICATION SYSTEM-ACCELERATED

A. ACCELERATED DETECTION

For TIDCS, the smaller the $TVSU$ is, the shorter the node status update is and the faster the malicious users can be distinguished. To improve the detection performance, we propose an enhanced novel scheme called Trust-based Intrusion Detection and Classification System- Accelerated (TIDCS-A). For instance, TIDCS takes into consideration the nodes' histories during a period of time. However, the network behavior is characterized by complexity and randomization. Consequently, TIDCS-A is proposed to verify and control the network behavior based on the correlation between the decisions made by the classifier that does not depend on any past information and the decisions of TIDCS-A generated based on the past performance of the nodes. The proposed scheme divides the time into multiple intervals γ. Let γ_i be an interval of γ with index i and size s. The network operator fixes s based on the network load, and according to the requirements (time/security level). Basically, in a dynamic network, s should be small as much as possible for a fast update to detect any change in the network. However, there is some standard closed cloud setting (.i.e. private cloud) where the network is more stable, s can have bigger values to avoid the last update. So, during each γ_i, the proposed system computes the decision similarity between the classifier and TIDCS-A decisions denoted by SMI which counts the number of times that the classifier and TIDCS-A have the same decisions classification. So the bigger is SMI, the more stable behaviors the network is. For γ_i, SMI can be written as:

$$SMI(\gamma_i) = \sum_{\gamma_i} \left(classifier(d_t^k) = TIDCS - A(d_t^k) \right). \quad (7)$$

Let δ be the accuracy ratio between the created classifier and TIDCS-A. δ computes the ratio of the similarity between the created classifier and TIDCS-A decisions during the interval γ_i. δ can be expressed as:

$$\delta(\gamma_i) = \frac{SMI(\gamma_i)}{s}. \quad (8)$$

We define the correlation rate μ_i as a new metric to improve the system detection performance. μ_i shows the degree of correlation between the classifier and TIDCS-A. μ_i is used

to control the nodes behaviors in time. So, at the beginning of the each interval γ_i, TIDCS-A uses the current and the last accuracy ratios ($\delta(\gamma_i)$ and $\delta(\gamma_{i-1})$ respectively) to compute μ_i and manages the windows past decision size *WPS*. The correlation rate μ_i can be written as:

$$\mu_i = \begin{cases} 1 & \text{if } i = 1, \\ \left| \dfrac{\delta(\gamma_i) - \delta(\gamma_{(i-1)})}{\delta(\gamma_{(i-1)})} \right| & \text{if } i > 1. \end{cases} \tag{9}$$

The decision of *WPS* is based on a threshold τ fixed according to the security levels, such that:

- **Case 1:**

$$\mu_i > \tau, \tag{10}$$

The decisions of the classifier and TIDCS-A diverge and only the last γ_i past decisions will be taken into consideration.

- **Case 2:**

$$\mu_i \leq \tau. \tag{11}$$

The decisions of the classifier and TIDCS-A converge and all the previous decisions will be taken into consideration.

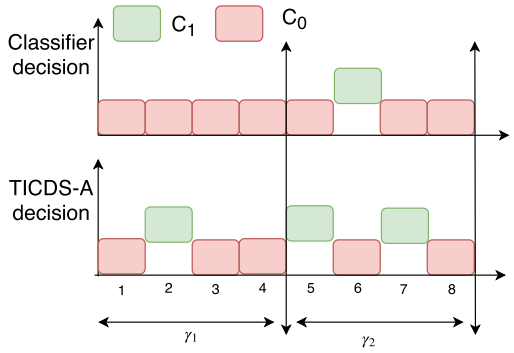

FIGURE 5. An example of the decisions of the classifier and TICDS-A for $\gamma = \{1, 2\}$.

Figure 5 presents an example of two intervals $\gamma = \{1, 2\}$ with size $s = 4$ (4 time unit (TU)). In this example, two classes c_0 and c_1 are considered. For γ_1, the similarity *SMI* is equal to 3, and $\delta(\gamma_1)$ is equal to:

$$\delta(\gamma_1) = \frac{3}{4} = 0.75.$$

For γ_2, the similarity *SMI* is equal to 1 since the classifier and TIDCS-A have only one common decision (when TU=8). In this case:

$$\delta(\gamma_2) = \frac{1}{4} = 0.25.$$

μ_2 is equal to:

$$\begin{aligned} \mu_2 &= \left| \frac{\delta(\gamma_2) - \delta(\gamma_1)}{\delta(\gamma_1)} \right|, \\ &= \left| \frac{0.25 - 1}{1} \right| = 0.75. \end{aligned} \tag{12}$$

In this example, and if we set $\tau = 0.5$, we have $\mu_2 > \tau$ (case 1) so only the last decision will be taken is c_0 which is the same decision of the classifier. The nodes' histories will be updated starting from $\gamma = 3$.

Consequently, an algorithm is presented in the next section to design an optimized *WPS* to guarantee the fast convergence of the TIDCS-A decisions. This algorithm is used to compute Eq 7, Eq 8 and control the *WPS* by satisfying the conditions in Eq 10 and Eq 11.

B. OPTIMIZED WINDOWS PAST INFORMATION

To compute the the similarity *SMI*, the accuracy ratio δ and μ, and to control the conditions in Eq 10 and 11, a novel best effort algorithm entitled *OptimizedWPS* (Alg 1) is designed.

Algorithm 1 *OptimizedWPS* (TR, TS)

1: Input:
2: TR : Training dataset.
3: TS : Testing dataset.
4: Output:
5: WPS : Optimized window past decision.
6: ———————————
7: OptimizedWPS.Initialization.
8: OptimizedWPS.FinalDecison.

The first step in the proposed algorithm consists of initializing the necessary variables using the function of *OptimizedWPS.Initialization* defined in Ag. (2). γ and τ are considered as input from the network operator.

Algorithm 2 *OptimizedWPS.Initialization* (γ, τ)

1: $s \leftarrow Length(\gamma)$. ▷ *The size of the input interval* γ.
2: $n \leftarrow Length(C)$. ▷ *Number of classes in* C.
3: $\mu \leftarrow 1$. ▷ *Initialize the accuracy correlation rate* μ *to 1.*
4: $SMI \leftarrow 0$. ▷ *Initialize the similarity rate to 0.*
5: $\delta \leftarrow 0$. ▷ *Initialize the accuracy ratio to 0.*
6: $WPS \leftarrow ComputeRatio(d_t^k, SMI, \delta)$. ▷ *Compute the initial WPS.*

The function $ComputeRatio(d_t^k, SMI, \delta)$ is a function created in Alg. 3. This function computes first the similarity *SMI* then the accuracy rate δ and the correlation rate μ for each interval γ_i and compares it to τ to see whether the decisions of the classifier and TICDS-A converge or not. In the worst case, the decisions diverge meaning that the node changes its status rapidly during γ_i and it is performing abnormal behaviors. In this case, the algorithm does not take into consideration all the past data of this node, and only the new decisions will be considered.

After finishing the computation of the *WPS*, the algorithm summarizes all the previous decisions r_i^k for each node k (Line 1 of Alg. 4) and updates the historical recurrence vector R_t^k according to the *WPS*. The algorithm computes the decision d_t^k (Line 3 of Alg. 4) that provides the index

Algorithm 3 *ComputeRatios* (d_t^k, SMI, δ)

Input: d_t^k, SMI, δ.

Ouput: WPS

$SMI(\gamma_i) \leftarrow \sum_{\gamma_i}(classifier(d_t^k) = TIDCS - A(d_t^k))$ ▷ *Number of similar decisions between the classifier and the TICDS-A fusing* γ_i.

$\delta(\gamma_i) \leftarrow \frac{SMI(\gamma_i)}{s}$ ▷ *Compute the accuracy ratio.*

$\mu_i \leftarrow \frac{\delta(\gamma_i)-\delta(\gamma_{(i-1)})}{\delta(\gamma_{(i-1)})}$ ▷ *Compute the accuracy correlation rate.*

if $(\mu_i < \tau)$ **then**

    $WPS \leftarrow (t - i)$ ▷ *The system considers all the previous decisions of the node from the received instant* t *to the last accurate decision in instant* i.

else

    $WPS \leftarrow t$ ▷ *The system considers only the decision in instant* t.

end if

Return (WPS)

Algorithm 4 *OptimizedWPS.FinalDecison.*

Input: WPS.

Ouput: d_t^k

for $i \leftarrow 1$ to N **do**

    $r_i^k \leftarrow \sum_{t=WPS}^{T} d_t^k = c_i$.

    Update the historical recurrence vector $R_t^k = [r_1^k \dots r_n^k]$.

    $d_t^k \leftarrow \underset{c_i}{argmax}(R_k^t(c_i))$

end for

Return(d_t^k)

of the maximum element in the vector R_t^k which is the final classification decision.

VII. PERFORMANCE EVALUATION

In this section, the performance of the proposed approach is studied. First, we present the simulation environment in section (VII.A) including the used machine learning algorithms. Section (VII.B) presents the used datasets. The feature selection performance section (VII.C) is investigated in terms of accuracy and detection rate ; false and positive rate. Also, we use precision, recall, and F1 function for better performance evaluation [8]. TICDS and TICDS-A performances are presented in sections (VII.C) and (VII.E) respectively and a comparison between the two models is studied in section (VII.F).

A. SIMULATION ENVIRONMENT

Without loss of generality, we use this approach on two machine learning algorithms namely: **decision tree** and **random forest (50 trees)**. Both algorithms are very well-known machine learning approaches and used in many classification problems. Moreover, they are suitable for large datasets which makes them optimal for network anomaly detection over cloud computing [37], [38]. We assume that each node changes its status every $500TU$. Note that the total number of nodes is fixed to 50 nodes proportionally distributed to the number of classes in the datasets.

B. USED DATASETS

1) NSL-KDD DATASET

In 2009, the NSL-KDD data has been released as a refined version of the KDD cup99, it includes 41 features and 4 attack categories which are DOS, PROBE R2R, and U2R. These attacks include 39 attack types.

The validation method is hold-out where the dataset is partitioned into two parts 70% from the testing set used to create the optimized score weights and 30% is used to compare the performance of the algorithms.

2) UNSW DATASET

NSL-KDD dataset is an upgraded version of the KDD99 dataset. The major disadvantage of this dataset is that it does not represent the modern low foot-print attack scenarios. To overcome the deficiencies of the old datasets, the UNSW-NB15 dataset was developed by the cyber-security research group at the Australian Center for Cyber Security in 2015. The dataset contains nine different modern types of attacks and varieties of real normal traffic. The data was generated with a change over time to imitate the contemporary real network traffic

The UNSW dataset [39] includes 10 different types of traffic packets and it is more suitable to be used in the contemporary anomaly detection models. It includes normal packets as well as 9 types of attacks, which are Analysis, Backdoor, DoS, Exploits, Fuzzers, Reconnaissance, Shellcode, and Worms. Table 2 shows the notation of these classes in the paper. UNSW-NB-15 is composed of two parts: a training set *UNSW-NB-15 training-set.csv* which has been used for model creation and a testing set, *UNSW-NB-15-testing-set.csv* used for the testing step and modeling the received real-time packets.

TABLE 2. Classes notation.

Number	Class
0	Normal
1	Analysis
2	Backdoor
3	DoS
4	Exploits
5	Fuzzers
6	Generic
7	Reconnaissance
8	Sellcode
9	Worms

C. FEATURE SELECTION PERFORMANCE

Figure 6 shows the redundancy and accuracy of each feature during the random phase for the decision tree

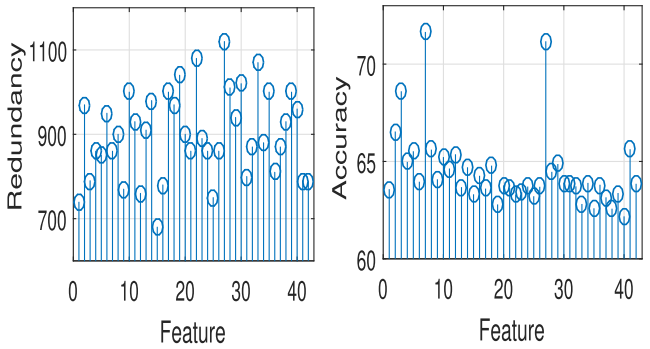

FIGURE 6. The accuracy and the redundancy for TIDCS applied to decision tree using UNSW-NB15.

using UNSW-NB15. The redundancy shows how many times a feature F participates in group generation (random phase) and the accuracy per feature shows the score that has been assigned to this feature. The redundancy of features varies between 700 and 1100 and the feature accuracy between 62 and 75%. First, we remark that the number of redundancy does not impact the accuracy since a bigger redundancy value does not mean a bigger accuracy rate. For example, in Figure 6, we can see that F_7, which has the highest accuracy rate compared to the other features, has a redundancy equals to 870 which is not the maximum. This means that important features can be identified from the first iterations. In this simulation, the number of iteration is $N_{training} = 1100$ which can be modified according to the network requirements in terms of complexity and time.

Figure 7 shows the accuracy during the random (a) and improvement (b) phases for TIDCS using UNSW-NB15. This figure shows how the system improves its performance after completing the training phase. The improvement phase shows the final performance of the TIDCS. During the random phase, the accuracy shows an excessive fluctuation which means that the best features are not identified and the generation of more groups results in variations of the accuracy. After completing the random phase, the system has enough information about each feature and it can identify the best

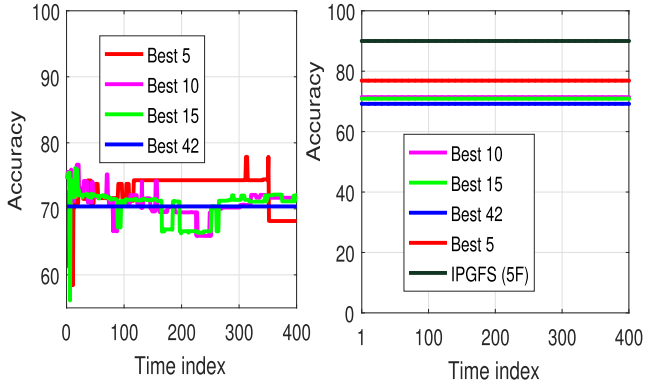

FIGURE 7. The accuracy rate of TIDCS applied to decision tree during the training and improvement phases using UNSW-NB15.

of them according to their given accuracy score. According to Figure 7, the best 5 features have higher accuracy (78%) than the 42 features (71%). The proposed features selection technique increases the detection accuracy by 9% compared to the original performance of the decision tree (69%) while reducing the number of features to 5 out of 42 (80% less). In this simulation, we show also the effect of the use of past data on system performance. We can remark that the accuracy reaches 91%. This means 13% improvement compared to the best 5 features model and 20 % compared to the original decision tree. In fact, after long periods of detection, the amount of nodes' past decisions also increases. Therefore, the system has enough information about node behaviors and can make more accurate decisions.

Figure 8 shows the accuracy rate of the proposed approach applied to the decision tree and bagging tree compared to the original algorithms using UNSW-NB15 datasets. This figure shows that the proposed model works with different machine learning algorithms TIDCS selects the best 5 features for both algorithms which proves that it can reduce the number of features whatever the algorithm is. The use of past data also improves the detection performance of the decision tree and bagging tree in terms of accuracy rate with 23% and 7% respectively compared to the original performance.

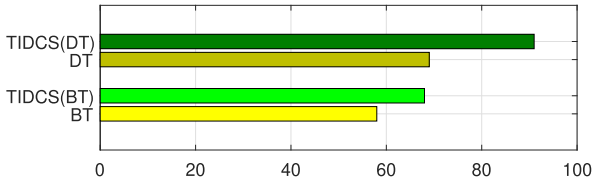

FIGURE 8. Accuracy of TIDCS applied to decision tree and bagging trees.

Table 3 shows the online time complexity of the representative machine learning algorithms [40]. n is the number of instances, each described by $m$attributes. Generally, n depends on the dataset size which cannot be managed. However, m which is the number of features can be managed by selecting the most important features. The experiments have been conducted in an operating system on a core $i7$ desktop computer with 16 GB RAM.

TABLE 3. Time complexity.

Algorithm	Time complexity	Comments
Decisions Trees	$O(m * n^2)$	m: attributes n: instances
Random Forest	$O(M * m * n * \log(n))$	M: Number of trees

Figure 10 shows the effect of the number of features m on the time complexity. We fixed $M = 500$, $n_1 = 67343$ (testing set of NSL KDD) and $n_2 = 175,341$ (testing set of UNSW-NB-15). m is varied from 5 to 42. First, we remark that the decision tree (DT) has a bigger complexity than random tree (RF). This figure shows that m has a strong effect on the time complexity. The bigger is m, the higher is the complexity.

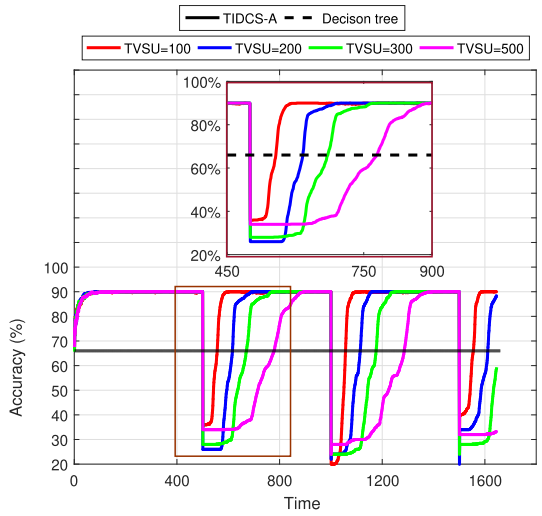

FIGURE 9. The accuracy rate of TIDCS applied to decision tree compared to the the original algorithm under different *TVSU* configurations as a function of time.

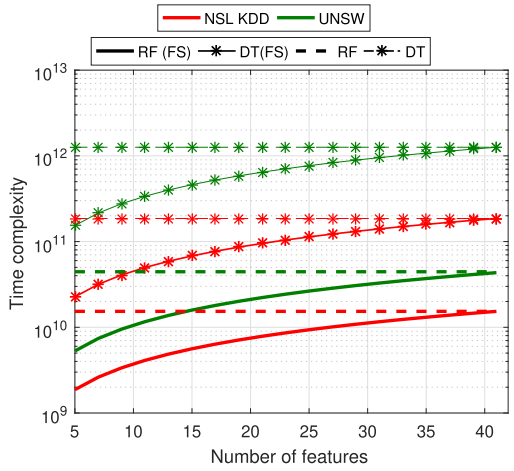

FIGURE 10. Time complexity.

The accuracy performance aims to select as many features as possible, while on the contrary, the overfitting problem of the created model requires to reduce the number of features as possible for low model complexity and precision.

Figure 11 depicts a comparison of performance between random forest (RF), Silent features (SF) and TIDCS applied to the decision tree using the NSL-KDD dataset. The results show that all the presented systems have good accuracy (>99%). However, this rate is achieved by TIDCS with only 5 features while RF and SF use 10 features.

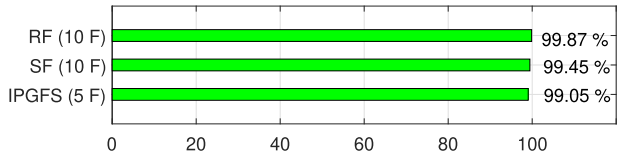

FIGURE 11. Comparison of performance between RF, SF and TIDCS using the NSL-KDD dataset.

Tables 5 and 4 a comparison of performance between RF, SF, and TIDCS using the UNSW-NB15 and NSL-KD datasets to select the most important features. TIDCS is applied to the decision tree and it reduces the original 42 UNSW-NB15 features to 5, and the original 41 NSL-KDD features to 5. We compare the results of TIDCS to SF and RF models. The three systems select different features that have a big impact on their accuracy performance.

TABLE 4. NSL-KDD feature selection.

FS technique	Number of features	Selected features
Original features	41	F1, F2, F3, F4, F5, F6,F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19,F20, F21, F22, F23, F24, F25,F26, F27, F28, F29, F30, F31,F32, F33, F34 , F35, F36, F37,F38, F39, F40,F41
Silent Features	10	F5, F23, F24, F29, F31,F33, F34, F35, F37, F39
Random Forest selection	10	F4,F3, F8, F5, F10, F23,F30, F32, F36, F40
TIDCS	5	F5, F3, F23, F35, F4

TABLE 5. UNSW-NB15 feature selection.

FS technique	Number of features	Selected features
Original features	42	F1, F2, F3, F4, F5, F6,F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19,F20, F21, F22, F23, F24, F25,F26, F27, F28, F29, F30, F31,F32, F33, F34 , F35, F36, F37,F38, F39, F40,F41,F42
Silent Features	5	F8, F25, F26, F29, F31
Random Forest selection	5	F23, F38, F37, F20, F4
TIDCS	5	F2, F3, F7, F27, F42

D. TICDS PERFORMANCE

1) SIMULATION RESULTS USING UNSW DATASET

Figure 9 shows the accuracy rate of TIDCS applied to decision tree compared to the original algorithm under different *TVSU* configurations as a function of time. We can see that by increasing the time (between 1 to 400), the accuracy of TIDCS increases however; the accuracy of the decision tree is fixed to 69%. In the zoomed part from Figure 9, the time varies between 450 and 900. First, we can remark that when time index=500, and $TVSU = 100$, TIDCS accuracy decreases to 30%. The node changes its status every $500TU$ so the past decisions of the nodes are not accurate which falsify the system decisions. After $100TU$ (time index=610) the system performance improves and reaches 91%. However, when $TVSU = 750$, TIDCS accuracy reaches 91% after 400 TU (time index=900). The smaller *TVSU* is, the more accurate the node past information will be.

Figure 12 presents the detection performance after 1, 400, 550, 900, 1050 and 1400 TU of TIDCS applied to decision tree. Note also that the normal node average decision is equal to 0 and the average for the malicious node is 1. The normal users are presented by blue stars and red stars for the malicious ones. The detection performance of the original decision tree is presented when the time index=1. We can remark that the decision tree has a weak detection performance meaning that it is enabled to distinguish the malicious nodes from the non-malicious ones. After 400 iterations, the proposed algorithm successfully distinguishes malicious users from the normal nodes by giving them low reputation scores. For Time=550, the system is not able to detect the malicious nodes because the participants' nodes change their status.

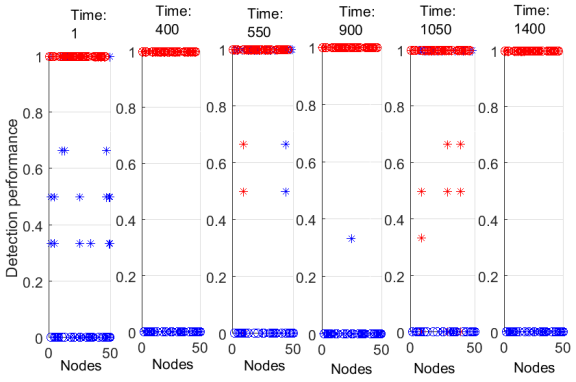

FIGURE 12. The detection performance after 1, 400, 550, 900, 1050 and 1400 TU for TIDCS applied to decision tree with $TVSU = 100$.

Figure 13 shows the accuracy rate, detection rate, and false-positive rate of TICDS compared to NB, AODE, CADF and TANN using UNSW-NB15 Dataset. It can be seen that the proposed system has a higher accuracy rate compared to the other techniques. For example, the accuracy for multi-classification using the UNSW-NB15 dataset is 91% for TICDS, 83.47% by using online AODE, 88% for CADF, 90% for EDM, 90% for TANN and 69.6% for NB. In addition, TICDS also provides the higher detection rate (94%) than TANN (88.2%), EDM (89.4%), AODE (77.84%) and NB(70.32%). Finally, for the false alarm rate, TICDS also performs the best (4%) over TANN (12.3%), EDM (10.6%), AODE (6.57%) and NB(31.67%).

2) SIMULATION RESULTS USING NSL-KDD DATASET

TANN shows good accuracy for the UNSW dataset. Thus, it has been selected to be compared with TICDS in terms of detection performance. Figure 14 shows the accuracy rate of TICDS compared to TANN. We can see that TANN keeps a good performance for intrusion detection and its accuracy reaches 96.91%. TIDCS also has a good accuracy equals to 98%. So, TIDCS has a better accuracy rate compared to TANN.

Table 6 shows a classification comparison between TIDCS ($TVSU = 500$) and TANN using NSL-KDD dataset. It can be

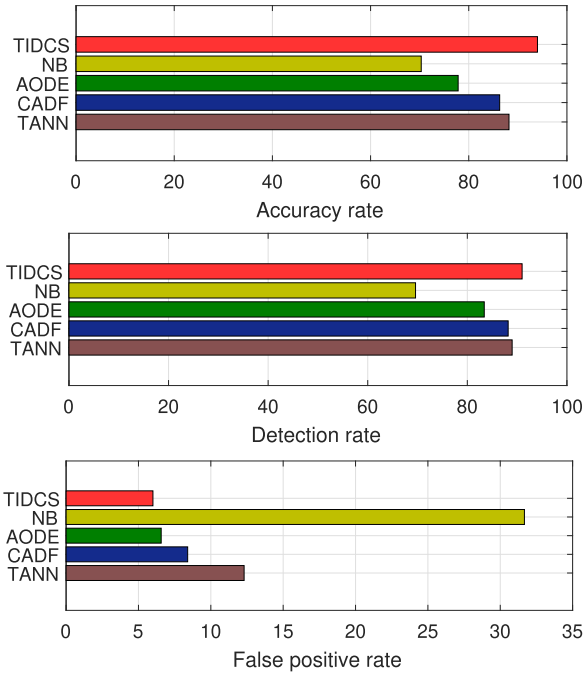

FIGURE 13. Accuracy rate, detection rate, and false positive rate of TICDS compared to NB, AODE, CADF and TANN.

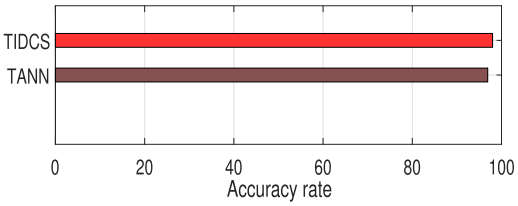

FIGURE 14. Accuracy rate of TICDS and TANN using NSL-KDD Dataset.

TABLE 6. Comparison between TIDCS and TANN using NSL-KDD Dataset.

	Normal	Probe	DoS	U2R	R2L
TANN	97.01	94.89	90.94	60	80.53
TIDCS	100	99	95	81	98

seen that we have small improvements in predictive accuracy for TIDCS compared with TANN.

E. TICDS-A PERFORMANCE

Figure 15 (a) shows the accuracy rate of accuracy rate of TIDCS-A applied to decision tree under different threshold τ with a fixed interval size $s = 10$. Similarly to TIDCS, the accuracy of TIDCS-A increases with time. When $TU = 500$, the TIDCS-A accuracy decreases, however, the system detects rapidly the changes in the first interval and updates the status using only the 10 past decisions. The accuracy increases from 30% to 91%. In the zoomed part from Figure 15, we can see the impact of τ on the accuracy which controls the decreased rate of accuracy value. For example, $\tau = 0.5$, the accuracy reaches 20% and for $\tau = 0.2$, it reaches only 30%. So the lower the τ is, the better the accuracy will be.

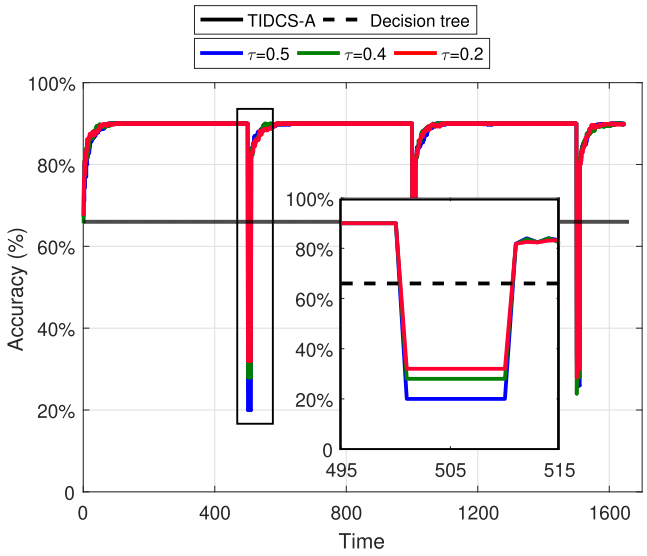

FIGURE 15. The accuracy rate of TIDCS-A applied to decision tree under different threshold τ with a fixed interval size $s = 10$.

F. TIDCS VS TIDCS-A

We assume that each node changes its status every 300TU.

Figure 16 shows a 3D figure for the accuracy rate per class for TIDCS as a function of time for $TVSU = 300$. First, we can remark that TIDCS has a 100% accuracy for 6 class classifications which are (0,4,5,6,7,8) and has a weak detection performance for the rest (1, 2, 3, 9). In particular, class 10 has 0 packets detected from 100 received. This weak performance is due to the deficiency of decision tree algorithm.

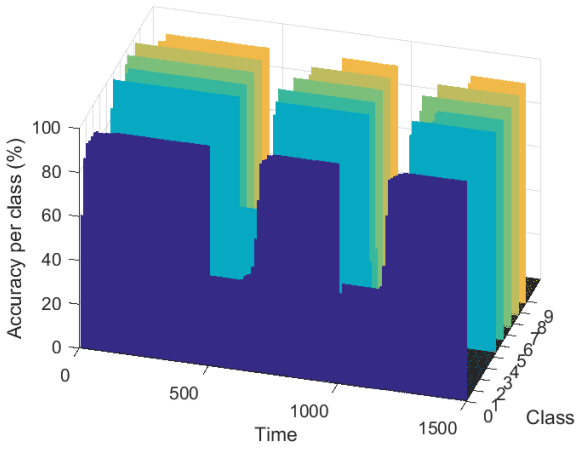

FIGURE 16. The accuracy rate per class for TIDCS for $TVSU = 300$.

Figure 17 shows the accuracy rate per class for TIDCS-A in function of time with $\tau = 0.5$ and $s = 10$. We can see that TIDCS-A detects the intrusion rapidly compared to TIDCS and the accuracy of classes reaches 100% faster than TIDCS. Thanks to its dynamic window update, TIDCS-A detects the evolution of the behaviors of nodes and updates *WPS* accordingly which increases the intrusion detection.

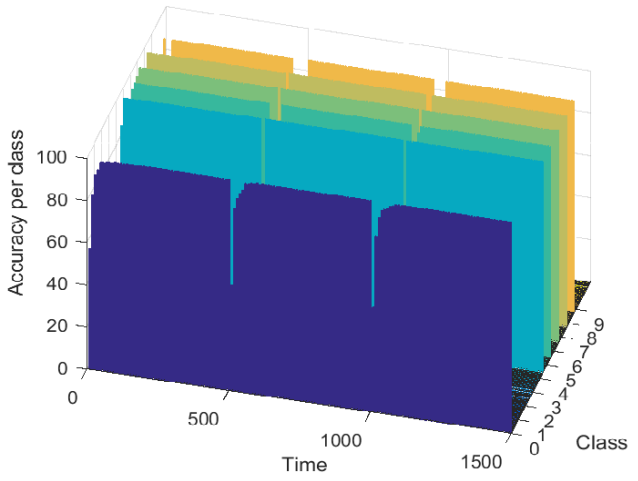

FIGURE 17. The accuracy rate per class for TIDCS-A in function of time with $\tau = 0.5$ and $s = 10$.

Table 7 shows the detection rate comparison for TIDCS and TIDCS-A applied to two machine learning algorithms namely: decision tree (DT)and random forest (RF) using the UNSW dataset. The size s of interval γ is equal to 10 and $\tau = 0.2$. *TVSU* is equal to 500. We remark that both systems TIDCS and TIDCS-A show good results in terms of detection accuracy. However, when time=500, the detection rate decreases for both models applied to decision tree (the nodes change their status every 500 TU). Only 100 TU are needed for TIDCS-A to improve its performance and the detection rate reaches 88% when time=600. On the other hand, the detection rate of TIDCS reaches 88% only when time=800. TIDCS-A has an optimized algorithm to identify *WPS* which improves the detection rate while increasing the time and computation complexity. However, TIDCS is more simple with a fixed cleansing period.

TABLE 7. The detection rate comparison between complex tree and bagging trees using TIDCS and TIDCS-A using UNSW dataset.

	TIDCS-A $s = 50$ τ=0.2		TIDCS TVSU=500	
Time	**DT**	**RF**	**DT**	**RF**
1	66	53	66	52
50	86	66	86	66
501	30	18	32	30
510	30	18	32	30
550	87	18	32	30
600	88	66	32	30
700	88	66	42	40
800	88	66	75	44
900	88	66	88	66

G. SUMMARY

Our preliminary investigation reveals that both TIDCS and TIDCS-A show good results in terms of detection accuracy. TIDCS applies periodic trust updates based on the node's past information. TIDCS-A proposes a dynamic algorithm to compute the exact time for nodes cleansing states and restricts the exposure window of the nodes. TIDCS and TIDCS-A are used according to the applied security policy. It has been proven that TIDCS-A is faster in the detection of malicious

nodes and more complex compared to TIDCS. By using the NSL-KDD and UNSW datasets, TICDS performs better than previous work (NB, AODF, CADF, and TANN) in terms of average accuracy, the detection rate, false alarm.

VIII. CONCLUSION

The security of the networks has become an essential issue in any distributed system. Intrusion detection systems came to aid in adding a layer of protection over these networks by detecting unauthorized intrusion scenarios. In this paper, we propose a novel model for network intrusion detection, namely TICDS and TICDS-A. In particular, the proposed system combines machine learning techniques and past information to create a trusted cloud environment. TICDS and TICDS-A apply cleansing activities for the participants' nodes, regardless of the presence/absence of attack alarms. TIDCS has a fixed periodic cleansing window and TIDCS-A has a dynamic window for network cleansing and anomaly detection. Simulation results show the good performance of the two proposed models.

REFERENCES

[1] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Comput. Netw.*, vol. 130, pp. 94–120, Jan. 2018.

[2] M. Aloqaily, S. Otoum, I. A. Ridhawi, and Y. Jararweh, "An intrusion detection system for connected vehicles in smart cities," *Ad Hoc Netw.*, vol. 90, Jul. 2019, Art. no. 101842.

[3] X. An, X. Lu, L. Yang, X. Zhou, and F. Lin, "Node state monitoring scheme in fog radio access networks for intrusion detection," *IEEE Access*, vol. 7, pp. 21879–21888, 2019.

[4] A. Aljumah and T. A. Ahanger, "Fog computing and security issues: A review," in *Proc. 7th Int. Conf. Comput. Commun. Control (ICCCC)*, May 2018, pp. 237–239.

[5] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, K. Khan, and N. Meskin, "Cybersecurity for industrial control systems: A survey," *Comput. Secur.*, vol. 89, Feb. 2020, Art. no. 101677.

[6] A. Hijazi and J.-M. Flaus, "A deep learning approach for intrusion detection system in industry network," Tech. Rep., 2019.

[7] I. E. Mir, D. S. Kim, and A. Haqiq, "Security modeling and analysis of an intrusion tolerant cloud data center," in *Proc. 3rd World Conf. Complex Syst. (WCCS)*, Nov. 2015, pp. 1–6.

[8] Z. Chkirbene, A. Erbad, and R. Hamila, "A combined decision for secure cloud computing based on machine learning and past information," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–6.

[9] I. Bilbao and J. Bilbao, "Overfitting problem and the over-training in the era of data: Particularly for artificial neural networks," in *Proc. 8th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2017, pp. 173–177.

[10] C. Lee, J.-H. Kang, and S.-P. Kim, "Feature slection using mutual information for EEG-based biometrics," in *Proc. 39th Int. Conf. Telecommun. Signal Process. (TSP)*, Jun. 2016, pp. 673–676.

[11] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, 2009.

[12] E. K. Tang, P. N. Suganthan, X. Yao, and A. K. Qin, "Linear dimensionality reduction using relevance weighted LDA," *Pattern Recognit.*, vol. 38, no. 4, pp. 485–493, Apr. 2005.

[13] A. S. Syed Navaz, T. D. Sri, and P. Mazumder, "Face recognition using principal component analysis and neural networks," *Int. J. Comput. Netw., Wireless Mobile Commun.*, vol. 3, pp. 245–256, Mar. 2013.

[14] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," *Knowl. Inf. Syst.*, vol. 34, no. 3, pp. 483–519, Mar. 2013.

[15] R. J. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," 2015, *arXiv:1509.07577*. [Online]. Available: https://arxiv.org/abs/1509.07577

[16] K. K. Gupta, B. Nath, and R. Kotagiri, "Layered approach using conditional random fields for intrusion detection," *IEEE Trans. Dependable Secure Comput.*, vol. 7, no. 1, pp. 35–49, Jan. 2010.

[17] M. Franke, A. Geyer-Schulz, A. W. Neumann, "Recommender services in scientific digital libraries," in *Multimedia Services in Intelligent Environments: Advanced Tools and Methodologies*, G. A. Tsihrintzis and L. C. Jain, Eds. Berlin, Germany: Springer, 2008, pp. 377–417, doi: 10.1007/978-3-540-78502-6_15.

[18] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014.

[19] J. Devi, J. Road, M. X. , and A. L. T. Size, "Dimensional data," in *Proc. Int. Conf. Inventive Syst. Control (ICISC)*, 2017, pp. 24–26.

[20] R. Sadoddin and A. Ghorbani, "Alert correlation survey: Framework and techniques," in *Proc. 2006 Int. Conf. Privacy, Secur. Trust: Bridge Gap Between PST Technol. Bus. Services*, New York, NY, USA, 2006, p. 37.

[21] A. K. Saxena, S. Sinha, and P. Shukla, "General study of intrusion detection system and survey of agent based intrusion detection system," in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, May 2017, pp. 421–471.

[22] G. Shang-fu and Z. Chun-lan, "Intrusion detection system based on classification," in *Proc. IEEE Int. Conf. Intell. Control, Autom. Detection High-End Equip.*, Jul. 2012, pp. 78–83.

[23] A. Borkar, A. Donode, and A. Kumari, "A survey on intrusion detection system (IDS) and internal intrusion detection and protection system (IIDPS)," in *Proc. Int. Conf. Inventive Comput. Informat. (ICICI)*, Nov. 2017, pp. 949–953.

[24] N. Moustafa, G. Creech, E. Sitnikova, and M. Keshk, "Collaborative anomaly detection framework for handling big data of cloud computing," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2017.

[25] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "Denial-of-service attack detection based on multivariate correlation analysis," in *Neural Information Processing*, B.-L. Lu, L. Zhang, and J. Kwok, Eds., Berlin, Germany: Springer, 2011, pp. 756–765.

[26] W. Wang, L. Chen, K. G. Shin, and L. Duan, "Secure cooperative spectrum sensing and access against intelligent malicious behaviors," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 1267–1275.

[27] C. Zina, M. Hasna, R. Hamila, and N. Hamdi, "Location privacy preservation in secure crowdsourcing-based cooperative spectrum sensing," *EURASIP J. Wireless Commun. Netw.*, vol. 2016, no. 1, p. 85, Dec. 2016.

[28] Z. Chkirbene, M. O. Hasna, R. Hamila, and N. Hamdi, "Energy-efficient based on cluster selection and trust management in cooperative spectrum sensing," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Apr. 2016, pp. 367–372.

[29] R. Zhang, J. Zhang, Y. Zhang, and C. Zhang, "Secure crowdsourcing-based cooperative pectrum sensing," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2526–2534.

[30] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 29, no. 4, pp. 462–472, 2017.

[31] B. A. Tama, M. Comuzzi, and K.-H. Rhee, "TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," *IEEE Access*, vol. 7, pp. 94497–94507, 2019.

[32] Y. Zhong, W. Chen, Z. Wang, Y. Chen, K. Wang, Y. Li, X. Yin, X. Shi, J. Yang, and K. Li, "HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning," *Comput. Netw.*, vol. 169, Mar. 2020, Art. no. 107049.

[33] C.-F. Tsai and C.-Y. Lin, "A triangle area based nearest neighbors approach to intrusion detection," *Pattern Recognit.*, vol. 43, no. 1, pp. 222–229, Jan. 2010.

[34] F. Gumus, C. O. Sakar, Z. Erdem, and O. Kursun, "Online naive bayes classification for network intrusion detection," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2014, pp. 670–674.

[35] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Multi-classification of unsw-nb15 dataset for network anomaly detection system," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 15, Aug. 2005.

[36] B. V. Nguyen, "An application of support vector machines to anomaly detection," Res. Comput. Sci.-Support Vector Mach., Tech. Rep. CS681, Aug. 2002.

[37] S. Sahu and B. M. Mehtre, "Network intrusion detection system using J48 decision tree," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Aug. 2015, pp. 2023–2026.

[38] R. M. Elbasiony, E. A. Sallam, T. E. Eltobely, and M. M. Fahmy, "A hybrid network intrusion detection framework based on random forests and weighted k-means," *Ain Shams Eng. J.*, vol. 4, no. 4, pp. 753–762, Dec. 2013.

[39] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.

[40] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.